REMARKS

Claims 1, 2, 4-12, 14-22, 24-28, and 30-41 were pending prior to this response. Claims 1, 10, and 28 were amended to include various limitations of dependent claims (i.e., cancelled Claims 9 and 30) and to better define the claimed invention (i.e., to add the limitation of an application instantiating the service connector based on a read configuration file in Claim 1 and to specify that the service connector is used to access network or distributed services rather than those in a local directory service in Claim 10) to place them in condition for allowance. Claims 37 and 38 were amended to provide clearer antecedent basis.

No new matter is added by these amendments with support found in the originally filed claims, Figures 3A and 3B, and the specification at least at page 18, lines 5-15 and page 19, 22-30). Claims 1, 2, 4-8, 10-12, 14-22, 24-28, and 31-41 remain for consideration by the Examiner.

Rejections Under 35 U.S.C. § 112

Claims 37 and 38 were rejected under 35 U.S.C. § 112, second paragraph, for insufficient antecedent basis. Claims 37 and 38 have been amended accordingly.

Rejections Under 35 U.S.C. § 103

In the Office Action of September 3, 2002, Claims 1-5, 7, and 9-36 were rejected under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 6,360,230 ("Chan"). This rejection was withdrawn in the Office Action dated December 19, 2002, but Claims 1, 2, 4, 5, 7-12, 14-22, 25-28, and 30-36 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Chan.

Claim 1 is directed to a method calling for "enabling definition of a service connector interface in conjunction with said first service" (i.e., the service being accessed) and then "subsequently invoking said service connector interface in conjunction with said second service...including instantiating said service connector interface at said second service." The method as amended clarifies that an application at the second service performs the invoking of the service connector interface and does so by "reading a configuration file providing an indicator for said service connector interface." Claim 1 requires that a service connector interface be defined for the service being accessed and that a configuration file be made available

to an application that then instantiates the service connector based on the configuration file.

The December 19, 2002 Office Action states that Chan fails to teach instantiating a service connector interface at the second service. It is then asserted that such implementation would be obvious in light of Chan. However, Chan merely teaches instantiating objects related to a directory service of a local environment. Chan describes how to uniformly access a number of directory systems one at a time but this requires the embedding of logic within the memory of the system (see Fig. 4, memory 401 with OleDs 408, 411, 414 that are provided for accessing directory services 409, 412, 415 and supporting text beginning at col. 8, line 62). The method of Claim 1, in contrast, includes steps that support dynamically referencing a generic service and to store and retrieve references to that service such that "it is not necessary to embed all of the logic necessary for the application 302 to determine how to get a reference directly from the service 306" (see, Applicant's specification at page 19, lines 1-30). To clarify this fundamental difference, Claim 1 calls for an application to perform the instantiating of the service connector interface by a configuration file that it first reads. Chan fails to teach reading a configuration file and then an application at a second service instantiating the indicated service interface connector at the second service. Applicant does not believe that it is obvious to modify the teaching of Chan to arrive at the method of Claim 1 nor that any motivation is provided to apply the teaching of Chan to a distributed environment and the interfacing with generic services available over a network. Hence, Claim 1 is believed allowable over Chan.

More particularly, Chan is directed to "a framework for uniform access to various different directory services" (see col. 6, lines 14 and 15). In the Chan system, "each provider of a directory service provides an implementation of these interfaces for their directory service that maps the behavior of their API set to the behavior of these OleDs interfaces. In this way, a client that is developed to use the OleDs architecture can access each of these directory services..." (see col. 6, lines 26-32) and the OleDs define a model "to assist clients in accessing the objects of a directory service" (see col. 7, lines 34 and 35). While providing some useful solutions, directory and meta-directory solutions "only help to catalog information that is already known and do not provide a mechanism for automatically discovering the identity and interface parameters for a network service" (page 5, lines 9-13 of Applicant's specification). Specifically, Chan does not teach invoking the service connector interface in conjunction with the second

service including instantiating the interface at the second service (and more particularly, instantiating based on a configuration file).

At col. 6, lines 37-47, Chan discusses "an interface for enumerating the directory services contained within the namespaces container" and that a client can discover the existence of directory services. The interface is not specific to a particular service (such as a first service) but is instead "an interface for enumerating the directory services contained within the namespaces container." After defining service connector interface for the first service, Claim 1 calls for the interface to be invoked in conjunction with the second service by instantiating at the second service. Chan at col. 7, lines 10-13 describes a binding function that invokes retrieved code but does not describe a second service (such as a client application) invoking the defined interface. Rather, Chan states that "the invoked code uses the API set of the directory service to access the identified object" without reference to an interface that is defined relative to a particular first service. Further, Chan does not teach instantiating the service connector interface at the second service. At col. 7, lines 14-25, a container object is instantiated but there is no teaching that such instantiation occurs at the second service (such as on a system running a client application in a distributed environment) or that the object is defined relative to the first service. Hence, Chan does not make Claim 1 obvious and unpatentable. Claims 2 and 4-8 depend from Claim 1 and are believed allowable for at least the reasons for allowing Claim 1.

In the Office Action of December 19, 2002, Claims 6 and 8 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Chan further in view of U.S. Patent No. 5,339,430 ("Lundin"). The rejection of these claims based on 103(a) is respectfully traversed based on the following discussion.

Claims 6 and 8 depend from Claim 1 which is believed allowable over the art of record, and therefore, Claims 6 and 8 are believed allowable at least for the reasons for allowing Claim 1. Further, Lundin fails to overcome the shortcomings of the teachings of Chan discussed above. Claims 6 and 8 are directed to the concept of defaulting to instantiating and then referencing a latest version or latest instance of a service being accessed or requested by client application or "second service." Lundin discusses using latest or new versions of software but does not discuss defaulting to such a choice **only upon a selection for a version or a particular instance not**

being provided by a requesting service as called for in Claims 5 and 7 from which Claims 6 and 8 depend. Hence, the combination of Chan with Lundin does not result in the method called for in Claims 6 and 8, and these claims are believed allowable over the art of record.

Independent Claim 10 is directed to a computer program product with some limitations similar to Claim 1 but further calls for gaining reference to the first service by the second service to include retrieving an instance of the first service at the service connector interface, obtaining a service reference from the first service, and then returning said reference to the second service. Further, Claim 10 as amended requires that the first service be in "a distributed environment" and that the second service be in "a local environment" and that "the service connector interface encapsulating logic necessary to retrieve service instances in the distributed environment not in a directory service in the local environment. Claim 10 is believed allowable for the reasons for allowing Claim 1 and also because Chan teaches directory service techniques within a local environment not obtaining generic services within from a distributed environment. Hence, Chan fails to teach that the first and second services are in different computing environments and that the service connector interface that is invoked within the local environment of the second service includes embedded or hidden logic necessary to obtain a reference to the first service in the distributed environment (i.e., the logic does not have to be embedded in an application in the local environment or in objects in memory at the second service). Independent Claim 10, and Claims 11, 12, and 14-18 that depend from Claim 10, are believed nonobvious in light of the teachings of Chan.

Independent Claim 19 is directed to a method similar to Claim 1 and is believed allowable for at least the reasons for allowing Claim 1 but further includes the limitation that a particular version of the first service can be specified by the second service. Chan fails to teach this added limitation. Specifically, at col. 7, lines 59-64, Chan simply refers to the idea of default namespace and directory service that allows a user to indicate what should be a default namespace for a directory service. The December 19, 2002 Office Action (in the Response to Arguments) states that "by defining a default namespace Chan clearly teaches the limitation of specifying a particular version." However, providing a namespace in the OleDs path or a default namespace in case there is nothing provided in the path **does not provide teaching of specifying the "version" of such code related to that namespace** as is required in Claim 19. Multiple

versions of the same underlying code may be available and Claim 19 calls for a method that allows a user to select the version of the first service to be invoked by the second service. Claim 19, and Claims 20-22 and 24-27 that depend from Claim 19, are not taught or suggested by Chan and are believed in condition for allowance.

As stated in the previous response, independent Claim 28 is directed to a system for providing dynamic references between services and is believed allowable for the reasons for allowing Claim 1 (as amended to include means for instantiating a service connector at the second service) but also because it calls for means for enabling definition of a service connector interface in conjunction with the first service that includes means for developing a computer program module adhering to the service connector interface in conjunction with the first service. As amended, Claim 28 is not taught or suggested by Chan and Claim 28, and Claims 30-32 and 34-36 that depend there from, are allowable over the art of record.

Also, in the Office Action dated December 19, 2002, Claims 37 and 39-41 were rejected under 103(a) as being unpatentable over U.S. Patent No. 6,026,404 ("Adunuthula"). This rejection is traversed based on the following remarks.

Independent Claim 37 is directed to a core profile engine as shown in Figure 2 with a pluggable interface for use in attaching to service modules and including a service connector associated with each attached service module. Claim 37 calls for "a pluggable interface attaching to the plug-in service modules" "wherein the pluggable interface further includes a service connector...adapted to receive the service request from the application programming interface and to return a reference to the one service module ..." The claimed service connector is not shown or suggested by Adunuthula and hence, Claim 37 is believed in condition for allowance.

The December 19, 2002 Office Action cites Adunuthula for teaching the service connector with its dispatcher 214 and that the reference is taught by a "handler" described at col. 15, lines 23-52. However, the dispatcher 214 provides a very different functionality than required of the service connector of Claim 37. With reference to col. 8, lines 1-37, the dispatcher 214 processes a browser request to authenticate the browser (if necessary) and identify a particular cartridge (i.e., a module of code for performing specific application or system function as defined at col. 6, lines 64-65) by communicating with a resource manager 254 to determine

which instance of the selected cartridge should be sent the request. The dispatcher 214 then "(4) creates and dispatches a revised browser request for execution by the specified instance of the cartridge."

The dispatcher 214 does not return a "reference to the one service module" as called for in Claim 37 (note, that Applicant could find no discussion of returning a reference to a requested service at col. 15, lines 23-52). Hence, Claim 37 and dependent Claim 38 are believed in condition for allowance. Claim 38 was also rejected based on the combination of Adunuthula and U.S. Patent No. 6,304,967 ("Braddy"), but Braddy does not overcome the deficiencies of Adunuthula with respect to base Claim 37. Claim 38 is believed allowable over Braddy when combined with Adunuthula.

Independent Claim 39 is directed to a method of providing an application with a reference to or access to a particular service. As discussed with reference to Claim 37, Adunuthula fails to teach a service connector as defined in the claims. The dispatcher 214 does not teach the service connector. There is not a teaching of instantiating the dispatcher 214 in relation to specific services based on an indicator read from a configuration file. Further, the dispatcher 214 does not return a reference to the service. At col. 12, lines 1-32, Adunuthula again discusses the idea of creating a modified browser request, communicating with a resource manager 254, if necessary even initiating new cartridges, and then sending the revised browser request to the selected cartridge. However, Adunuthula does not teach operating the dispatcher to identify an instance of a service, obtaining a reference to such service, and then returning the reference of the service to the application. Since each step of this method is not shown or made obvious by Adunuthula by itself or in combination with other references, independent Claim 39 is believed in condition for allowance. Claims 40-41 depend from Claim 39 and are believed allowable for at least the reasons for allowing Claim 39.

Conclusions

Based on the above remarks, all pending claims are believed to be allowable over the above-discussed references. Consequently, the case is believed to be in condition for allowance, and this action is respectfully requested.

No fees are believed due with this response, but any fee deficiency associated with this submittal may be charged to Deposit Account No. 50-1123.

Respectfully submitted,

Date 1/14/03

Kent A. Lembke, Reg. No. 44,866
Hogan & Hartson LLP
One Tabor Center
1200 17th Street, Suite 1500
Denver, Colorado 80202
Telephone: (720) 406-5378
Fax: (720) 406-5301

Version With Marking to Show Changes

IN THE CLAIMS:

Claims 1, 10, 28, 37, and 38 were amended as follows:

1. (Twice Amended) A method for providing a reference to a first service to a second service in a computer system comprising:

enabling definition of a service connector interface in conjunction with said first service;

subsequently invoking said service connector interface in conjunction with said second service, wherein said subsequently invoking includes instantiating said service connector interface at said second service; and

gaining reference to said first service by said second service[.] ;

wherein said steps of subsequently invoking said service connector interface and gaining reference to said first service are carried out by an application program operative in conjunction with said second service, the instantiating including reading a configuration file providing an indicator for said service connector interface.

10. (Twice Amended) A computer program product comprising:

a computer usable medium having computer readable code embodied therein for providing a reference to a first service in a distributed environment to a second service in a local environment in a computer system comprising:

computer readable program code devices configured to cause said computer system to effect enabling definition of a service connector interface in conjunction with said first service, the service connector interface encapsulating logic necessary to retrieve service instances in the distributed environment not in a directory service in the local environment;

computer readable program code devices configured to cause said computer system to effect subsequently invoking said service connector interface in conjunction with said second service; and

computer readable program code devices configured to cause said computer system to effect gaining reference to said first service by said second service, wherein said computer readable program code devices comprise:

computer readable program code devices configured to cause said computer system to effect retrieving a service instance at said service connector interface;

computer readable program code devices configured to cause said computer system to effect obtaining a service reference from said first service; and

computer readable program code devices configured to cause said computer system to effect returning said service reference obtained from said first service to said second service.

28. (Twice Amended) A system for providing dynamic references between services in a computer system comprising:

means for enabling definition of a service connector interface in conjunction with said first service, said definition enabling means includes means for developing a computer program module adhering to said service connector interface in conjunction with said first service;

means for subsequently invoking said service connector interface in conjunction with said second service, wherein said means for subsequently invoking said service connector interface comprises means for instantiating said service connector at said second service; and

means for gaining reference to said first service by said second service.

37. (Amended) A core profile engine for use in gateway or firewall servers for enabling client applications to access plug-in service modules in a distributed computing environment without embedding location and negotiation logic within the client applications, the engine comprising:

an application programming interface in communication with the client applications and adapted with interfaces for processing a request for a service provided by one of the plug-in service modules; and

a pluggable interface attaching to the plug-in service modules, wherein the attaching includes providing an initialization parameter comprising a storage location for each of the plug-in service modules;

wherein the pluggable interface further includes a service connector associated with each of the attached plug-in service modules that is adapted to receive the service request from the

application programming interface and to return a reference to the one service module providing the service based on the storage location.

38. (Amended) The core profile engine of claim 37, wherein the plug-in service modules are selected from a group consisting of an authorization plug-in, an authentication plug-in, a notification plug-in, a log plug-in, a group plug-in, an entity identification factory plug-in, and a replication plug-in.